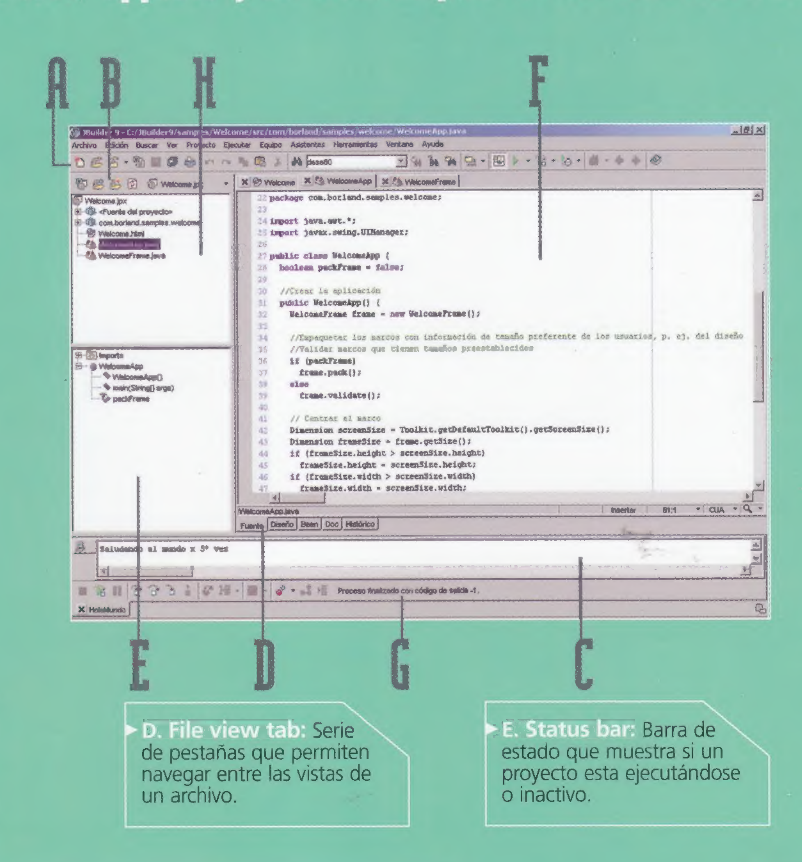
JBuilder PERSONAL EDITION

BORLAND

PC WORLD PERÚ

Entorno multiplataforma de desarrollo visual con el cual es posible crear aplicaciones empresariales, servlets, applets y JavaBeans para el entorno Java 2.



A. Toolbar: Ofrece los diferentes botones para un rápido acceso a funciones como crear un archivo, guardarlo, imprimirlo, etc.

B. Project Pane: Muestra el árbol del proyecto respetando las relaciones jerárquicas entre los archivos que lo conforman. Permite movernos a través de estos y ver su contenido (en el caso de los paquetes).

C. Message Pane: Muestra, entre otras cosas, la salida que se verá desde la consola. Este panel aparece solamente cuando compila, ejecuta o depura un proyecto.

D. File view tab: Serie de pestañas que permiten navegar entre las vistas de un archivo.

E. Status bar: Barra de estado que muestra si un proyecto esta ejecutándose o inactivo.

F. Content Pane: Área de edición. Visualiza lo que contenga el archivo seleccionado en el Project Pane.

G. Structure Pane: Panel que permite navegar a través de los métodos y variables miembro del archivo fuente seleccionado, los paquetes incluidos, los conflictos de comentarios javadoc y los errores de compilación.

H. Project toolbar: Barra de herramientas desde la que podemos hacer operaciones como añadir archivos existentes a nuestro proyecto, eliminar archivos, y cambiar fácilmente a otro de nuestros proyectos.

PRELIMINARES

- Esta ficha práctica referencia a la versión Personal 9 de JBuilder. Con respecto a la versión anterior, las diferencias más importantes son:
  - Mejoras en la productividad de muchas de las herramientas.
  - Mayor personalización del IDE.
  - Creador de recopilatorios más flexible.
  - Integración completa con Borland Optimize It Suite. Esto permite mejorar mucho la velocidad, fiabilidad y escalabilidad de la aplicación durante la fase de desarrollo.

- Además de los instaladores se necesitará de una llave ("key"), que permita finalizar la instalación del software. Dicha llave puede ser encontrada en: www.borland.com/products/downloads/download_jbuilder.html .

- Entre otras cosas, este IDE cuenta con un editor de código muy flexible en cuanto a formato, un depurador gráfico, un diseñador para aplicaciones visuales que ofrece diversos controles, asistentes rápidos e intuitivos, ejemplos de aplicaciones, y una amplia documentación, que va desde tutoriales básicos hasta ayudas rápidas.

- Como se explicó anteriormente, para instalar cualquiera de las versiones de este IDE, hace falta un archivo llave. Desde el link dado, debe elegirse la llave correspondiente a la versión del IDE que se desea instalar. Una vez seleccionado, pedirá un correo electrónico al cual será enviada como archivo adjunto. Este archivo debe ser guardado en la ruta home del usuario (este archivo trae algunas indicaciones sobre donde guardarlo, según el sistema operativo que se esté utilizando). Finalmente, ejecutar el instalador del software. En caso de que no se esté seguro de cuál es la ruta del home, JBuilder da la posibilidad de indicarla manualmente para completar la instalación.

CREANDO UN PROYECTO

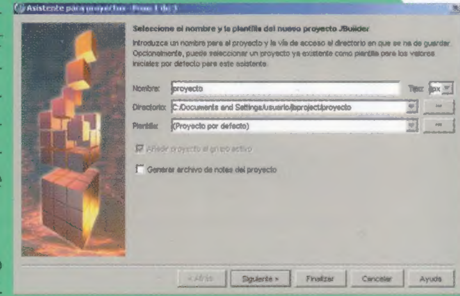
Antes de empezar con la programación propiamente dicha, es importante señalar que toda aplicación que se desarrolle debe estar dentro del entorno de un proyecto.

Para crear un proyecto, seleccione *File•New Project (Archivo•Proyecto nuevo)*. Aparece una ventana en la cual puede crearse un proyecto en tan solo tres pasos:

▸ Se elige el nombre del proyecto y el directorio donde se almacenará. Opcionalmente, se puede hacer un check sobre la selección *Generate project note files (Generar archivos de notas del proyecto)*, que crea un archivo con formato HTML, que contendrá algunas notas del proyecto.

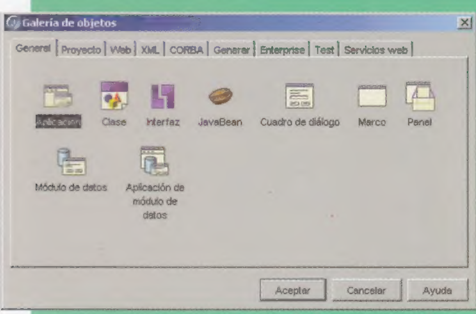
▸ Elija el lugar donde se almacenarán los archivos de salida y la versión del JDK que desea utilizar. De forma predeterminada se crean unas carpetas debajo de la carpeta seleccionada para el proyecto. Tome nota de estos datos ya que podría necesitar seleccionar algún archivo del proyecto de forma individual.

▸ Especifique algunas opciones generales del proyecto. La más resaltante aquí es la ventana de *Class Javadoc Fields (Archivos Javadoc de clases)*, en la cual pueden colocarse algunas descripciones del proyecto (esto será incluido en la página con formato HTML que genera el paso 1).



▸ Para cargar un proyecto guardado, seleccione *File•Open Project (Archivo•Abrir proyecto)*. Los archivos que levantan todo el proyecto normalmente tienen extensión ".jpx". Además es posible crear un proyecto nuevo, a partir de un entorno de archivos ya creados.

AÑADIENDO, ELIMINANDO Y BORRANDO ARCHIVOS DEL PROYECTO



▸ Para añadir archivos nuevos (como clases, interfases, etc.), seleccione *File•New (Archivo•Nuevo)*. Esto abre una ventana que permite añadir archivos de cualquier tipo al proyecto. Entre otros, puede elegirse:

▸▸ *Application (Aplicación)*: Genera los archivos básicos para una aplicación visual.

▸▸ *Class (Clase)*: Genera un archivo para una sola clase.

▸▸ *Interface (Interfase)*: Genera un archivo para crear una interfase.

▸▸ *JavaBean*: Implementa una clase JavaBean, que de modo predeterminado desciende de una clase JPanel, pero es posible extenderlo a cualquier componente.

▸▸ *Dialog box (Cuadro de diálogo)*: Implementa una clase para crear un cuadro de diálogo básico.

▸▸ *Frame (Marco)*: Implementa una clase para crear un contenedor del tipo marco.

▸▸ *Panel (Panel)*: Implementa una clase para crear un contenedor del tipo panel.

▸▸ *Data module (Módulo de datos)*: Genera los archivos necesarios para implementar una clase módulo de datos, que servirá para realizar conexiones a una base de datos.

▸▸ *Data module application (Aplicación de módulo de datos)*: Genera una aplicación para manipular el módulo de datos.

▸ En la pestaña *Web* además, se puede lanzar el asistente para la creación de applets. En la pestaña *Projects (Proyecto)*, se puede crear un proyecto nuevo; y, finalmente, en la pestaña *Generate (Generar)* se puede crear un repositorio. El resto de opciones no están disponibles para la versión Personal, a la que corresponde esta ficha práctica.

▸ JBuilder cuenta con un asistente bastante útil e intuitivo para la creación de cada tipo de archivo. Se puede también añadir archivos existentes al proyecto de forma rápida con el botón *Añadir archivos/paquetes/clases* ubicado en la barra de herramientas del proyecto.

▸ Para crear una aplicación visual, luego de abrir el asistente para creación de archivos y de haber seleccionado *Aplicación*, aparece una ventana que nos guía en la creación de la misma, que estará conformada por una clase que contendrá el método *main* y una clase *Frame* que contendrá las características de la ventana que está creando. En primer lugar se elige el nombre de la clase que contendrá el método *main* (opcionalmente puede seleccionarse el checkbox *Generate header comments, (Generar comentarios de cabecera)*, para generar automáticamente comentarios javadoc en la parte inicial del código fuente).

▸ Luego se elige el nombre de la clase *Frame* además de algunas de las características visuales que desee incluir en el marco, por ejemplo, una barra de menú, una barra de herramientas, etc.

▸ Por último, elige la configuración que será tomada como predeterminada para la compilación y ejecución del proyecto. No es necesario cambiar nada aquí para una aplicación básica.

▸ Para eliminar algún archivo del proyecto basta con seleccionarlo en el panel del proyecto, y luego presionar el botón correspondiente en la barra de herramientas del mismo. Si además desea borrarlo, puede hacerlo seleccionando el archivo, haciendo clic derecho y eligiendo la opción *Delete filename (Borrar nombre_de_archivo)* del menú contextual.

COMPILANDO Y EJECUTANDO EL PROGRAMA

Compilar es el proceso de transformar el código fuente. En el caso especial de Java se transforma a bytecodes, un punto intermedio que hace posible que las aplicaciones sean multiplataforma con la ayuda de la máquina virtual, y generar de este modo los archivos .class ejecutables.

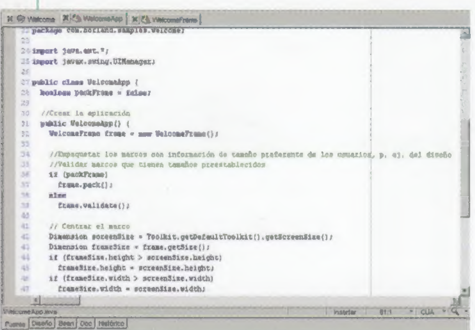
▸ Algunos tipos de archivos cuando son añadidos traen consigo una configuración predeterminada de compilación y ejecución, entre estos se encuentra el de tipo aplicación. Si este es el elegido, para compilarlo solo tendrá que ir al menú principal y elegir la opción *Run•Run project (Ejecutar•Ejecutar proyecto)*. Esto compilará y ejecutará el proyecto.

▸ Sin embargo, si por ejemplo, se añadió solamente un archivo de tipo clase al proyecto, hay que elegir la configuración que luego se tomará como predeterminada para el mismo. Para esto, seleccione *Run•Run Project (Ejecutar•Ejecutar proyecto)* del menú principal, tras lo cual aparece un asistente de configuración.

▸ En este cuadro de diálogo, de modo predeterminado aparece seleccionada la pestaña *Run (Ejecutar)*. Al presionar el botón *New (Nuevo)*, aparece una ventana en la cual se puede elegir el nombre con el que se reconocerá a la configuración que se está realizando, y el archivo del proyecto que contiene el método "main". Una vez terminada esta tarea, el IDE tomará esta configuración cada vez que necesite que el proyecto sea compilado.

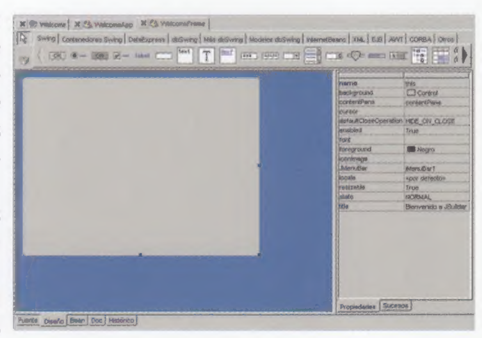
▸ En caso de que haya errores de sintaxis, estos aparecerán en el panel de mensajes, y el proyecto no se ejecutará hasta que sean resueltos. En el panel puede verse la ubicación del error en el código fuente y su posible causa. Es posible irse directamente hasta el error haciendo doble clic sobre el mismo.

►►► PROGRAMANDO



► La programación propiamente dicha se hará a través del *Content Pane,* lugar en donde por defecto, aparece el código fuente. Mediante las pestañas ubicadas en la parte baja de este panel *(File view tab)* puede moverse por las distintas vistas de un archivo: *Source (Fuente), Design (Diseño), Doc* (Documentación), *History (Historia)* y *Bean (Semilla).* La versión Enterprise soporta además la vista UML que muestra el modelo lógico del proyecto.

► La vista de diseño es soportada solo por algunos archivos, por aquellas clases contenedoras como las *Frame* y las *Panel*. Para editar un archivo en esta vista, se debe seleccionar del panel del proyecto el archivo adecuado, y luego elegir la pestaña *Diseño* en el *Content Pane*.



► JBuilder cuenta con una amplia gama de controles y contenedores prediseñados, que pueden ser incluidos en las ventanas de manera gráfica y sencilla. Existen desde controles que vienen incluidos en las librerías swing y awt, así como controles para bases de datos creados por Borland. En este sentido, se cuenta con una herramienta muy potente al tener un paquete de controles lo suficientemente variado como para desarrollar cualquier aplicación que se necesite. Para añadir algún control, debe seleccionarlo primero de la lista de controles, y luego seleccionar el área sobre el *Content Pane* que desea que ocupe, haciendo un clic y arrastrando el mouse.

name	this
background	Control
contentPane	contentPane
cursor	
defaultCloseOperation	HIDE_ON_CLOSE
enabled	True
font	
foreground	Negro
iconImage	
JMenuBar	jMenuBar1
locale	<por defecto>
resizable	True
state	NORMAL
title	Bienvenido a JBuilder

Propiedades | Sucesos

► En esta vista se cuenta, además, con una ventana de control de propiedades y eventos, que permite administrar fácilmente las características visuales de los controles, y las acciones que se tomarán como respuesta ante ciertos sucesos que se den (por ejemplo, un clic, una tecla presionada, etc.). Las propiedades visuales son editables haciendo clic sobre la que desee modificar. Para implementar la respuesta ante un evento debe cambiar primero a la pestaña *Events (Sucesos)* de la ventana de *Control*, y luego hacer doble clic sobre alguno de ellos. Esto aumentará el código necesario para programar la respuesta ante el evento, por lo que automáticamente se cambiará la vista del archivo al fuente, y el cursor se posicionará donde debe escribir el código.

► En la vista del código fuente puede ver un editor de código bastante flexible en cuanto al formato que quiera darle al mismo. Permite, entre otras cosas, configurar el teclado para atajos de teclas; controlar el formato del código, en su sangrado y tabulación; opciones de resaltado de llaves correspondientes; y presentación de errores. Para entrar a configurar estas opciones seleccione del menú principal la opción *Tools•Editor options (Herramientas•Opciones del editor)*.
En la pestaña "doc" puede ver la documentación generada de modo automático mediante el javadoc, en formato HTML. Recuerde que dicha documentación puede ser ampliada comentando debidamente en el código fuente.

►►► DEPURANDO

► Depurar es el proceso de detectar, diagnosticar y corregir problemas y fallas, revisando paso a paso un código fuente. El tipo de fallas que se suelen corregir corresponden al nivel lógico y de ejecución, mas no al nivel sintáctico. Una falla de nivel lógico se puede reconocer cuando las acciones que se realizan durante la ejecución del programa no son exactamente las que el programador tenía en mente. Por ejemplo, los errores lógicos pueden producirse cuando las variables contienen valores incorrectos, cuando los gráficos no son los adecuados, o cuando los datos producidos por el programa son incorrectos.

► Los errores de ejecución se dan cuando, por ejemplo, el programa intenta abrir un archivo que no existe o dividir un número por cero. Aunque este tipo de errores, en gran parte, son manejados por las excepciones en la parte de programación.

► Toda esta tarea de depuración se hace básicamente con dos acciones, deteniendo la ejecución del programa en ciertos puntos del código, y colocando puntos de inspección en los cuales se desea revisar los valores de variables o estado de los objetos.



► Para depurar un programa bastará con seleccionar del menú principal la opción *Run•Debug Project (Ejecutar•Depurar proyecto)*. El programa podría compilar el código antes de empezar con la depuración. A continuación aparecerán una serie de botones debajo del panel de mensajes. Dichos botones permiten controlar el código para la tarea de depuración.

► Lo siguiente que debe hacerse es asignar puntos de interrupción o "breakpoints", los cuales tienen la función de detener la ejecución del programa en

```
    if (packFrame)
      frame.pack();
    else
      frame.validate();

    // Centrar el marco
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height)
      frameSize.height = screenSize.height;
    if (frameSize.width > screenSize.width)
      frameSize.width = screenSize.width;
```

WelcomeApp.java

el lugar en el que son puestos. Para colocar uno de éstos, basta con hacer un clic sobre el área gris que se encuentra al costado izquierdo del área de edición del código fuente, a la altura de la línea en la que desea detener la ejecución del programa.

►►► DEPURANDO

► Otro modo es seleccionando *Run•Add breakpoint (Ejecutar•Añadir punto de inspección)*, y luego se elige el tipo de punto de interrupción que se desea. La tercera opción es presionando el botón respectivo en la barra para depuración . Si elige interrupción por línea, aparecerá una ventana solicitándo el nombre de la clase y la línea de código en la que desea colocar el punto de interrupción.

► Una vez colocados estos puntos, tal vez pueda necesitarse conocer el valor de algunas variables al momento de la detención del programa. Para esto, necesita volver a ejecutar el depurador y añadir puntos de observación o inspección. Esto último puede hacerse sombreando la variable que desea seguir, y con el botón derecho abrir el menú contextual para seleccionar *Add watch (Añadir punto de observación)*. Inmediatamente aparecerá un cuadro de diálogo con el nombre de la variable que se eligió. Opcionalmente puede colocarse una descripción de la variable. Otra forma de añadir un punto de inspección es seleccionando *Run•Add watch (Ejecutar•Añadir punto de observación)*, o con el botón respectivo en la barra de depuración .

► Si se vuelve a presionar el botón para reanudar la depuración , y luego el botón para ver las variables inspeccionadas en la barra de depuración, se notará que en la ventana de mensajes aparecerán dichas variables con sus respectivos valores. El botón ubicado en la barra de depuración detendrá totalmente la ejecución del programa.
JBuilder, a través de la opción *Help*, incorpora tutoriales para las diferentes fases de creación y compilación de proyectos.

►►► UTILIZANDO EL JAVADOC

► Javadoc es una herramienta creada por Sun Microsystems para generar documentación en archivos con formato HTML a partir de comentarios específicos en el código fuente. Para ver esta documentación elíjase la pestaña doc en el *Content Pane*. Se pueden añadir comentarios Javadoc a una clase e interfase, así como comentarios a un método, constructor y campo.

► Básicamente, la herramienta Javadoc extrae los comentarios Javadoc de los archivos fuente, y los coloca en los archivos de documentación con formato HTML. Un comentario Javadoc empieza con un símbolo de comienzo de comentario (/**), y termina con un símbolo de final de comentario (*/). Cada comentario consta de una descripción seguida de una o más etiquetas. Si lo desea, puede utilizar formato HTML dentro de los comentarios Javadoc.

► El siguiente es un ejemplo de cómo crear comentarios javadoc para una clase y un método:

```
import java.io.*;
/**
*Aquí debería colocarse una explicación de la utilidad de la clase
*@author nombre del autor de la clase
*/
public class CreadorPlantillas {
    /**
    *Aquí una explicación de lo que hace este método
    *@param parametro1 breve comentario sobre la variable "parametro1" que
    * estamos pasándole al método aumentarOpciones
    *@param parametro2 y aquí para la variable "parámetro2"
    */
    public void aumentarOpciones (int parametro1, float parametro2){
    ...
    }
}
```

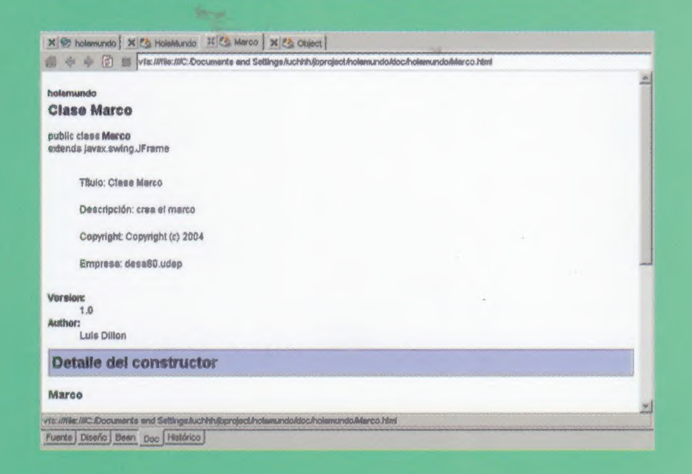
► La documentación generada a partir de la documentación del método se vería:

metodoAumentar
public void aumentarOpciones(int parametro1,
float parametro2)
Aquí una explicación de lo que hace este método

Parameters:
parametro1 - breve comentario sobre la variable "parametro1" que estamos pasándole al método aumentarOpciones
parametro2 - y aquí para la variable "parámetro2"

► Como puede verse la palabra @param es una etiqueta prediseñada que, en este caso, sirve para incluir valores bajo el título *Parameters* que muestra la lista de argumentos que se pasan a los métodos. Existen otras etiquetas prediseñadas, entre las más útiles se tienen:

ETIQUETA	DESCRIPCIÓN
@author	Nombre del autor de un código
@value	Comentarios sobre el valor de una constante
@return	Comentario sobre la variable retornada por un método
@throws	Comentario sobre una excepción



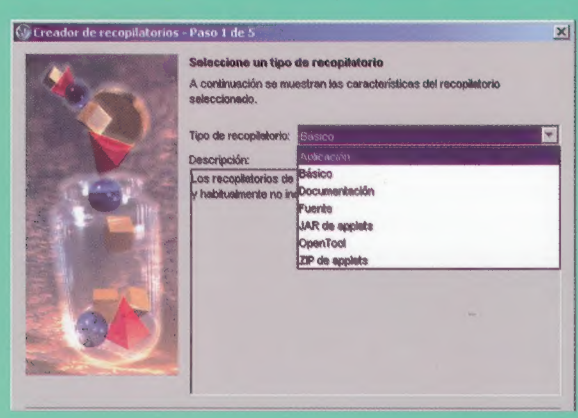
► Podrían producirse algunos errores al momento de generar la documentación Javadoc. Dichos conflictos pueden verse en el panel que muestra la estructura del proyecto, en la agrupación *Javadoc conflicts (Conflictos javadoc)*.

► El tutorial sobre Javadoc del JBuilder, que se puede acceder a través de *Help•Building Applications with JBuilder•Building Applications•Creating Javadoc from source files" (Ayuda•Construyendo aplicaciones con JBuilder•Construyendo aplicaciones creando Javadoc desde archivos fuente)*, muestra entre otras cosas, la lista completa de las etiquetas "@etiqueta". También se cuenta con la información publicada en la página web de Sun, www.java.sun .com/j2se/javadoc/

CREANDO EJECUTABLES Y RECOPILATORIOS

JBuilder cuenta con un creador de recopilatorios, un asistente que permite agrupar los archivos y paquetes de un proyecto en un contenedor, para luego poder distribuirlo con facilidad a otros usuarios. Se puede crear una amplia gama de recopilatorios como, por ejemplo, applets, aplicaciones, clientes de aplicaciones J2EE, archivos recopilatorios JAR ejecutables, documentación, archivos fuente, aplicaciones web, y archivos recopilatorios JAR de aplicaciones. Aunque la versión Personal no lo permite, es posible crear también ejecutables nativos de nuestros proyectos.

Para acceder al asistente creador de recopilatorios, seleccione *Wizards•Archive builder (Asistenes•Creador de recopilatorios)*. Mostrará un cuadro de



diálogo mediante el cual podrá crear los distintos tipos de recopilación vistos antes. Para efectos didácticos tomaremos como ejemplo la creación de un recopilatorio de aplicación, el cual puede ser creado en 6 pasos:

1. Elija este tipo de recopilatorio, *Application (Aplicación)*.

2. Elija el nombre del nodo sobre el cual se almacenarán los archivos resultantes del proceso de recopilación (meramente referencial); además, puede elegir el nombre del archivo de salida (con extensión ".jar").

3. Elija todos los archivos que se incluirán en el recopilatorio. Por defecto aparecerá la opción *All classes and resources (Todas las clases y recursos)*.

4. En el cuarto y quinto paso, al estar creando un recopilatorio simple, no se modificará nada. Déle clic al botón *Next (Siguiente)*.

5. Elija la clase principal del proyecto, es decir, aquella que contiene al método *main*.

6. Hecho esto, en el panel del proyecto aparecerá un nodo con el nombre que haya especificado. Dé un clic con el botón derecho del mouse sobre el nodo para que aparezca el menú contextual y seleccione *Generate (Generar de nuevo)*. Ahora tendrá la aplicación completa en un solo archivo, que puede ser ejecutada mediante el comando *jar –nombreArchivo*.

OTRAS OPCIONES

Es posible añadir, eliminar y configurar atajos de teclas. Seleccione del menú principal *Herramientas•Opciones del IDE*. En la pestaña *Visualizador*, el botón *Personalizar* muestra un cuadro de diálogo con todas las teclas rápidas configuradas. Podrían presentarse algunos conflictos si se está utilizando el escritorio KDE en Linux.

JBuilder posee asistentes que sirven para implementar interfases y modificar métodos de clases ya implantadas de modo rápido y sencillo. Para acceder a estos asistentes basta con seleccionar la opción *Wizards (Asistentes)* del menú principal, y luego elegir el asistente que se desee (para Interfases o Métodos).

CONECTÁNDONOS A UNA BASE DE DATOS

Una aplicación de base de datos es aquella que accede a datos almacenados y permite visualizarlos, así como modificarlos o manipularlos si fuera necesario.

En la mayoría de los casos, los datos se almacenan en bases de datos diseñadas expresamente para ese fin. Sin embargo –y esto no es menos válido–, también se pueden almacenar en archivos de texto (o con otro formato). La diferencia estará marcada por las ventajas que presentan estas bases de datos –con muchas funciones comunes disponibles para el programador– sobre el tratamiento que se debe hacer en el caso de trabajar con archivos de texto con nuestro formato particular.

JBuilder permite acceder y manipular la información que se encuentre almacenada en una base de datos haciendo uso de las propiedades, los métodos y los sucesos definidos en los paquetes DataSet de la biblioteca de componentes DataExpress, en combinación con el paquete dbSwing.

- Cuando una aplicación solicita información de una fuente de datos, como una base de datos, se denomina aplicación cliente.

- Cuando una aplicación gestiona solicitudes de datos de varios clientes se denomina aplicación servidor, y se le conoce comúnmente como un Sistema de gestión de bases de datos (SGBD).

Las aplicaciones JBuilder se comunican con servidores de bases de datos a través de la API JDBC, que es la especificación de conectividad con bases de datos de Sun. JDBC es la API Java estándar para acceso y manipulación de información en bases de datos. Las aplicaciones de bases de datos JBuilder logran conectarse con cualquier base de datos que tenga un controlador JDBC.

DataExpress es un paquete de clases e interfases de Borland que permiten el acceso básico a datos. Este paquete también define clases básicas de proveedor y almacén, así como una clase abstracta de DataSet que se extiende a otros objetos DataSet. Estas clases así definidas posibilitan el acceso a la información almacenada en bases de datos y otras fuentes de datos.

Una aplicación simple para conectarse a una base de datos se puede lograr con tres componentes básicos:

- Los componentes Database
- Los componentes QueryDataset
- El paquete dbSwing

Los componentes Database son específicos de JDBC y gestionan las conexiones con JDBC. Para acceder a datos utilizando un componente

CONECTÁNDONOS A UNA BASE DE DATOS

QueryDataSet, deberá asignar a la propiedad DataBase del mismo un componente Database que haya sido instanciado. Una base de datos puede estar compartida por varios conjuntos de datos, siendo este modo de trabajo lo que sucede más a menudo.

▸ **Un componente QueryDataSet** es un DataSet específico de JDBC que gestiona un proveedor de datos de JDBC, como se define en la propiedad query. Mediante el uso de un componente QueryDataSet en JBuilder se puede extraer datos de una fuente de datos a un componente StorageDataSet. Este proceso se denomina suministro. Una vez suministrados los datos, pueden visualizarse y utilizarse localmente en componentes enlazados a datos. Los componentes QueryDataSet utilizan sentencias SQL para acceder o suministrar datos desde la base de datos.

▸ **El paquete dbSwing** permite generar aplicaciones de base de datos que aprovechan la arquitectura de los componentes Swing de Java. Además de las subclases enlazadas a datos que incorporan de origen la mayoría de los componentes Swing, dbSwing incluye varias utilidades diseñadas específicamente para el desarrollo de aplicaciones basadas en DataExpress y JData Store. En la siguiente lista se indican algunos de los componentes dbSwing disponibles en la pestaña dbSwing de la paleta de componentes:

- TableScrollPane
- JdbTable
- JdbNavToolBar
- JdbStatusLabel
- JdbTextArea
- JdbComboBox
- JdbLabel
- JdbList
- JdbTextPane
- JdbTextField

▸ **dbSwing** ofrece importantes ventajas frente a Swing, como una mayor variedad de funciones y la posibilidad de enlazar con datos. Además, db Swing es ligero, su aspecto visual puede simular el de muchas plataformas diferentes y cumple estrictamente los estándares de Swing. Si desea obtener más información sobre el paquete dbSwing, consulte la documentación API.

SUMINISTRO DE DATOS DE LOS EJEMPLOS.

A continuación se explica la forma de configurar una aplicación de base de datos sencilla. La consulta que se va a utilizar en estos ejemplos es: SELECT *FROM CUENTA. Esta sentencia SQL selecciona todas las columnas de una tabla denominada CUENTA de una base de datos ficticia ubicada en un servidor de base de datos local.

▸ Para configurar una aplicación para su uso con los ejemplos:

▸▸ Cree una aplicación visual como se ha explicado anteriormente

▸▸ Active el diseñador de interfaces de usuario, seleccionando la pestaña Diseño.

▸▸ Haga clic en el componente Database de la pestaña DataExpress de la paleta de componentes, y a continuación haga clic en el árbol de componentes o en el diseñador de interfases de usuario para añadir el componente a la aplicación.

▸▸ Abra el editor de la propiedad Connection del componente Database creado, seleccionándolo y haciendo doble clic en el botón puntos suspensivos de la propiedad connection en el Inspector. Asigne valores a las propiedades de conexión: La URL de conexión apunta al host donde se encuentra la base de datos; en nuestro caso, base de datos local. Se debe seleccionar además el nombre de usuario y la contraseña con la cual deseamos conectarnos.

▸ El cuadro de diálogo Connection contiene un botón Probar conexión. Púlselo para comprobar que las propiedades de conexión tienen los valores correctos. Los resultados del intento de conexión se muestran en el área de estado. Cuando la conexión sea satisfactoria, pulse Aceptar.

▸▸ Para poder realizar una consulta, añada un componente QueryDataSet al diseñador, haciendo clic en el componente QueryDataSet de la pestaña DataExpress, y después haga clic en el árbol de componentes. Seleccione en el Inspector la propiedad query del componente QueryDataSet , pulse el botón de puntos suspensivos para que se abra el cuadro de diálogo QueryDescriptor y asigne valores a las siguientes propiedades:

PROPIEDAD	VALOR
Base de datos	nombreBaseDatos
Sentencia	SELECT * FROM CUENTA

▸ Haga clic sobre Probar consulta para cerciorarse de que se ejecuta correctamente. Cuando en el área de estado se indique Correcto, cierre el cuadro de diálogo pulsando Aceptar.

▸ Para ver los datos en la aplicación, añada los siguientes componentes a la interfase y asócielos al conjunto de datos como se explica a continuación:

▸▸ Seleccione contentPane (BorderLayout) y asigne a su propiedad layout el valor null .

▸▸ Coloque un JdbNavToolBar en la zona de la parte superior del panel, en el diseñador de interfases. Dicho componente se asocia automáticamente al objeto DataSet que tiene el foco, por lo que no es necesario definir su propiedad dataSet.

▸▸ Coloque un JdbStatusLabel en la zona de la parte inferior del panel, en el diseñador de interfases. Este componente se asocia automáticamente al objeto DataSet que tiene el foco, por lo que no es necesario asignar valores a su propiedad dataSet.

▸▸ Añada un componente TableScrollPane de la pestaña dbSwing al centro del panel, en el diseñador de interfases.

▸▸ Coloque un componente JdbTable en el centro del componente tableScrollPane creado anteriormente y asigne a su propiedad dataSet como valor el nombre del componente queryDataSet. Puede observar que en este momento el diseñador muestra datos dinámicos.

▸▸ Seleccione Ejecutar•Ejecutar proyecto para ejecutar la aplicación y examinar el conjunto de datos. En la etiqueta de estado de esta aplicación se ve el número de registros visibles. Cuando la aplicación se ejecuta por primera vez, en la etiqueta de estado se puede leer Registro 1 de "N" (donde "N" muestra el número de filas recuperadas en el conjunto de datos de cada aplicación).

▸ **DIRECTOR:** BERNARDO ROCA REY MIRÓ QUESADA ▸ **EDITORA:** FRANCA CAVASSA ▸ **EDITORA ADJUNTA:** MELISA OSORES ▸ **DESARROLLO:** MANUEL MASÍAS
▸ **GERENTE COMERCIAL:** AUGUSTO TOWSEND ▸ **GERENTE DE PRODUCTO:** RENZO MARIÁTEGUI ▸ **JEFE DE PRODUCTO:** PAOLA CORNEJO
▸ **COORDINACIÓN:** DORA NIQUEN ▸ **EDITORA DE DISEÑO:** CLAUDIA BURGA-CISNEROS ▸ **DISEÑO:** VERUZKA NORIEGA ▸ **DIAGRAMACIÓN:** GERARDO CRISTÓBAL
▸ **EDICIÓN, PREPRENSA E IMPRESIÓN:** EMPRESA EDITORA EL COMERCIO S.A., PERÚ, AÑO 2004